

Mettre en œuvre un système

Dossier de conception

-

ESE A

Table des matières

1. Introduction	2
2. Gestion de projet.....	3
2.1 Cadre de travail.....	3
2.2 Gestion du code (GitHub).....	4
3. Base De Données	5
3.1 PhpMyAdmin	5
3.2 BDD Optiplant	5
4. Interface Web.....	11
4.1 Gestion des utilisateurs.....	11
4.2 Interactions avec les bacs	13
4.3 Interactions avec les recettes	14
5. CSS	15
5.1 Design du site web	15
5.2 Esthétique et Accessibilité	16
6. API Météo	19
6.1 Contexte et objectif.....	19
6.2 Fonctionnalités et structure	20
6.3 Installation et utilisation	20
6.4 Données météorologiques et traitement	20
6.5 Conclusion	20
7. Livestream	20
8. Mise en service LattePanda	22
8.1 Ouverture des ports	22
8.2 Installation des logiciels nécessaires	22
8.3 Configuration de phpMyAdmin	23
8.4 Configuration de MySQL pour l'accès distant	23
8.5 Accès à la LattePanda depuis le réseau local	23
8.6 Documentation et support	24
9. Conclusion	24

1. Introduction

Ce projet s'inscrit dans la continuité de la SAE (Situation d'Apprentissage et d'Évaluation) réalisée par les étudiants en Automatique et Informatique Industrielle (AII) au cours du semestre 5. Lors de cette SAE, les étudiants ont travaillé sur la partie automatisation d'une ombrière, en développant des fonctionnalités telles que la gestion des volets pour réguler la lumière, l'arrosage automatisé des plantes, ainsi que la récupération d'énergie via des panneaux photovoltaïques. Ces éléments ont permis de poser les bases d'un système intelligent et partiellement automatisé.

Cependant, pour rendre ce système entièrement autonome et supervisable, plusieurs améliorations et ajouts sont nécessaires. Actuellement, il n'existe aucun outil de supervision permettant de monitorer l'état du système en temps réel, ce qui limite son efficacité et sa praticité. Pour pallier ces lacunes, plusieurs fonctionnalités doivent être intégrées, notamment :

- Une API météo pour enrichir les données environnementales et optimiser les décisions du système.
- Une fonction de Livestream permettant de visualiser en temps réel les bacs de plantes et de capturer des images pour créer des time lapses.
- Une interface web dotée de comptes utilisateurs, offrant un accès centralisé aux informations météorologiques (issues de l'API et de la station météo de l'ombrière), aux recettes d'alimentation des plantes, ainsi qu'aux données relatives aux différents bacs.
- Une base de données pour stocker et organiser l'ensemble des informations utiles au bon fonctionnement du système.

Ces ajouts permettront non seulement de rendre le système plus autonome, mais aussi de le rendre accessible et supervisable à distance, ouvrant ainsi la voie à une gestion plus efficace et intuitive de l'ombrière.

La figure suivante montre un schéma global du système.

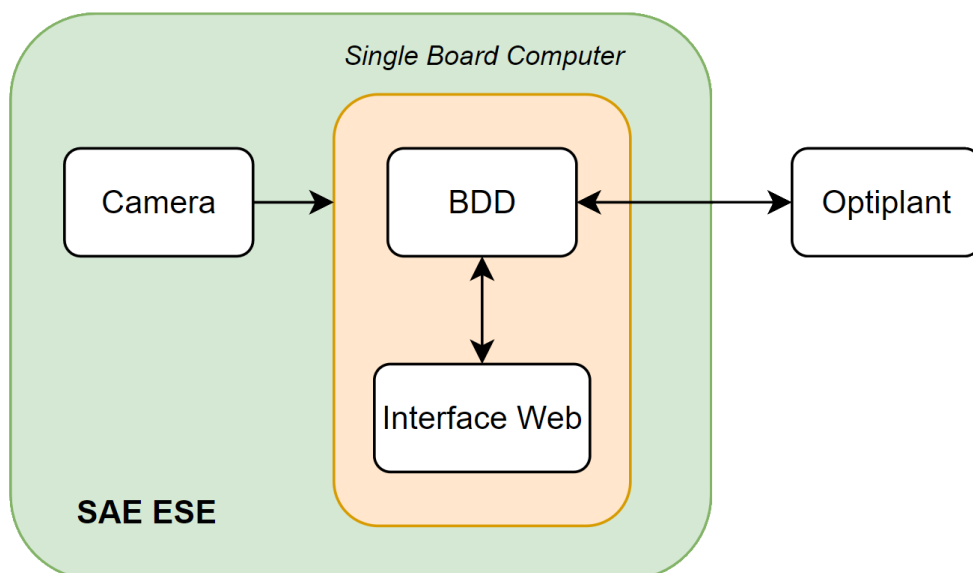


Figure 1 – Schéma global du système

2. Gestion de projet

La réalisation de ce projet constituant la SAE de semestre 6 est en rupture, tant au niveau de la forme que du contenu, avec les travaux réalisés précédemment au cours du BUT. En effet, il se différencie au niveau de la forme avec le nombre de personnes impliquées et par le contenu, car la majorité des compétences mobilisées ont à trait avec la programmation Web. À ce titre, la mise en place d'une gestion de projet est primordiale.

Dans les faits, la mise en place de cette gestion s'est déroulée en plusieurs phases :

- Le découpage du projet en plusieurs sous-catégories, comme décrit dans l'introduction
- Le positionnement des membres du groupe dans chacune de ces catégories
- La mise en place d'une méthode de travail de type SCRUM simplifiée

2.1 Cadre de travail

Afin de coordonner les membres du groupe qui travaillent pour la plupart sur des fonctionnalités interdépendantes, le développement de l'application a été séparé en un ensemble de tâches chacune caractérisé par une description, une "deadline" et une personne assignée.

L'approche choisie pour gérer ce projet est largement utilisée dans le développement de produit complexe ou d'application. Ainsi, il existe de nombreux outils permettant de trier, récapituler et gérer les différentes tâches, comme Trello par exemple.

Toutefois, une approche plus ancienne a été choisie pour synthétiser l'ensemble des tâches : un tableau avec trois colonnes, contenant chacune des post-it caractérisant chacune des tâches, représenté ci-dessous :

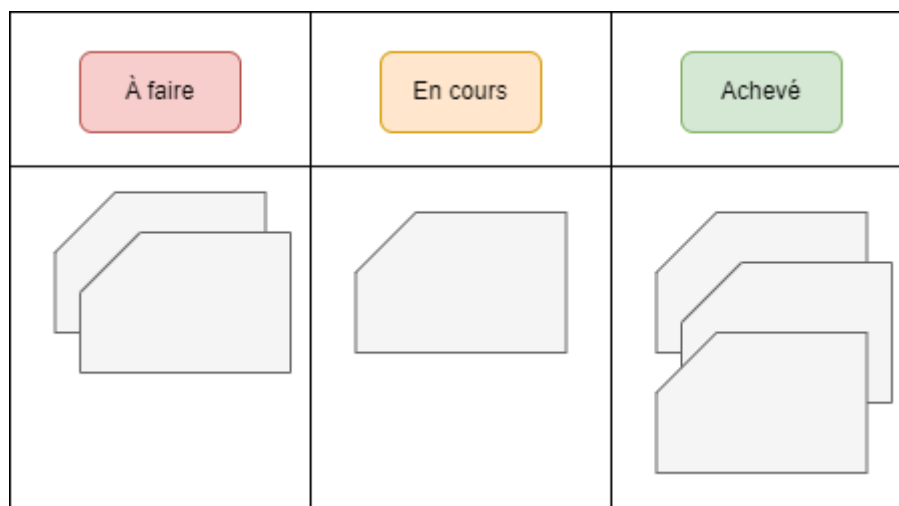


Figure 2 – Schéma de principe du tableau de synthèse des tâches

D'aucuns pourraient considérer l'utilisation d'un tableau blanc à défaut d'une alternative dématérialisée comme arriéré et en hors du temps. Cependant, elle dispose de plusieurs avantages jugés ici, particulièrement intéressants.

Dans un premier temps, son existence matérielle incite les parties prenantes du projet à y porter attention. En effet, le tableau est au centre de la salle de travail et à la vue de tous. Ainsi, il est naturel pour chacun de s'y référer. Le principal objectif de cette approche est de garantir que chaque membre du projet utilise l'outil choisi.

De plus, l'expérience dans le cadre universitaire montre que nombre de listes "TODO" ou "README", au nom pourtant évocateur, sont très rarement lus. Ainsi, l'utilisation d'un outil matérielle maximise l'implication de chacun dans l'entretien de celui-ci. Plusieurs facteurs peuvent expliquer ce constat : par exemple l'aspect ludique et la simplicité.

Toutefois, il est clair de ce genre de pratique ne s'applique que dans le cadre universitaire et ne pourrait pas être maintenu efficacement dans un environnement professionnel. Une solution dématérialisée comme évoqué plus haut serait alors à préférer.

2.2 Gestion du code (GitHub)

La gestion des versions du code est un autre aspect primordial du projet. Pour ce faire, l'outil Git couplé avec GitHub a été choisi. Git permet à une équipe potentiellement très nombreuse de collaborer sur un même projet et de garder en mémoire les modifications.

Avant ce projet, beaucoup n'avait qu'une connaissance de surface de Git et de GitHub qui correspondait aux attentes des projets universitaires antérieurs. Toutefois, le grand nombre de personnes travaillant en parallèle à nécessiter l'application de bonne pratique inhérentes à l'utilisation de Git pour la collaboration notamment l'utilisation des branches.

Un workflow classique avec Git repose sur la création de branches pour chaque nouvelle fonctionnalité ou correction de bug. Cela permet d'isoler les modifications et de travailler de manière indépendante sans affecter la branche principale, souvent nommée *main* ou *master*. Par exemple, lorsqu'un développeur souhaite implémenter une nouvelle fonctionnalité, il crée une nouvelle branche à partir de la branche principale. Cette branche, nommée *f-<nom-de-la-fonctionnalité>*, devient l'espace de travail dédié à cette tâche. Une fois le développement terminé et testé, la branche est fusionnée (merge) dans la branche principale via une pull request sur GitHub. Cette étape permet de revoir le code, de discuter des modifications et de s'assurer que tout fonctionne correctement avant l'intégration.

En parallèle, la branche principale doit toujours rester stable et fonctionnelle. Pour cela, il est essentiel de ne jamais pousser directement des modifications sur cette branche, mais de toujours passer par des branches dédiées. En cas de bug détecté, une branche de hotfix peut être créée à partir de la branche principale pour corriger rapidement le problème. Une fois le correctif appliqué, cette branche est fusionnée à la fois dans la branche principale et dans les branches de développement en cours pour éviter toute régression.

Ce workflow, bien que simple, garantit une gestion efficace du code source, une collaboration fluide entre les membres de l'équipe et une stabilité du projet tout au long de son développement.

Dans ce projet, le *repository* "Interface-Web" contenant le code de l'application représente 223 commits dans la branche *main* qui représente un total de 6 204 lignes de codes incluant les commentaires.

3. Base De Données

3.1 PhpMyAdmin

PhpMyAdmin est un outil open-source qui permet de gérer les bases de données (BDD) MySQL et MariaDB à travers une interface web graphique. Cet outil est simple d'utilisation, offrant une gestion intuitive des bases de données. PhpMyAdmin permet de créer, modifier, supprimer des bases de données, gérer des tables, et exécuter des requêtes SQL complexes. De plus, il permet d'importer et d'exporter les BDD simplement pour permettre à chaque membre de récupérer et d'utiliser cette BDD.

Dans ce projet, il est utilisé avec WAMP Serveur, un environnement de développement local, phpMyAdmin devient encore plus accessible. WAMP (Windows, Apache, MySQL, PHP) est une plateforme qui permet d'installer un serveur web complet sur un ordinateur Windows, incluant les composants Apache, MySQL, et PHP. L'intégration de phpMyAdmin dans WAMP permet ainsi une gestion fluide des bases de données directement depuis un navigateur web. Cela permet de modifier et de tester la BDD en local.

3.2 BDD Optiplant

3.2.1 Pourquoi notre système a besoin d'une base de données

Notre système d'ombrières automatisées avec gestion de recettes et de bacs de plantations nécessite une base de données robuste pour répondre à plusieurs besoins essentiels. Premièrement, la quantification précise des données constitue un élément fondamental pour assurer le bon fonctionnement et l'optimisation de notre installation. Nous devons suivre en temps réel les paramètres environnementaux (température, humidité, luminosité), l'état des cultures (croissance, santé, rendement), la consommation des ressources (eau, nutriments) ainsi que les calendriers d'entretien et de récolte. Cette masse d'informations requiert un système organisé capable de collecter, stocker et traiter efficacement les données.

La capacité de gestion qu'offre une base de données est indispensable pour notre projet. Elle nous permet d'automatiser les opérations quotidiennes, de gérer les alertes en cas d'anomalies, et d'orchestrer les différentes composantes du système. Grâce à elle, nous pouvons programmer les séquences d'irrigation, ajuster l'ouverture des ombrières selon les conditions météorologiques, et adapter les apports nutritifs en fonction des besoins spécifiques de chaque culture. Cette centralisation de la gestion garantit une cohérence dans les actions menées et optimise les performances globales du système.

Le stockage sécurisé et la modification aisée des données représentent également des avantages considérables. Notre base de données conserve l'historique complet des opérations, permettant ainsi des analyses rétrospectives et l'établissement de tendances. Elle offre également la flexibilité nécessaire pour modifier les paramètres, ajuster les recettes de culture, ou mettre à jour les protocoles de maintenance sans perturber le fonctionnement général. Cette adaptabilité est cruciale pour un système qui doit évoluer avec les saisons et s'améliorer continuellement.

3.2.2 Liste des données du système et leur interconnexion

Notre système génère et utilise de nombreuses données qui, grâce à une base de données relationnelle, peuvent être structurées en tables interconnectées pour une gestion optimale :

Les données environnementales comprennent les relevés de température, d'humidité, d'ensoleillement, de pluviométrie et de qualité de l'air. Ces paramètres influencent directement la gestion des ombrières et les cycles d'irrigation.

Les données relatives aux cultures incluent les informations sur les espèces cultivées, leurs stades de croissance, leurs besoins spécifiques, les rendements attendus et les historiques de production.

Les données techniques concernent l'état des équipements (ombrières, systèmes d'irrigation, capteurs), les calendriers de maintenance, les dysfonctionnements éventuels et les interventions réalisées.

Les recettes de culture regroupent les protocoles optimaux pour chaque type de plante, détaillant les conditions environnementales idéales, les cycles d'irrigation, les apports nutritifs et les soins particuliers à apporter.

Les données de gestion des ressources suivent la consommation d'eau, d'électricité et de nutriments, permettant d'optimiser l'utilisation des ressources et de réduire les coûts opérationnels.

Grâce à notre base de données, toutes ces informations sont non seulement stockées de manière structurée, mais aussi reliées entre elles par des relations logiques. Par exemple, les données environnementales sont associées aux recettes de culture pour déclencher automatiquement des actions sur les ombrières et les systèmes d'irrigation. Les informations sur les cultures sont liées aux calendriers d'entretien pour planifier les interventions nécessaires. Cette interconnexion des données permet une vision globale et cohérente du système, facilitant la prise de décision et garantissant une gestion optimale de notre installation d'ombrières automatisées.

3.2.3 Objectif et structure de la base de données

Enfin, l'objectif principal de la BDD Optiplant est de regrouper toutes les données nécessaires à la bonne gestion de l'ombrière afin qu'elle puisse être utilisée pour le site web. Les agriculteurs et gestionnaires peuvent ainsi surveiller à distance l'état de leurs cultures, consulter les historiques de production, ajuster les paramètres de fonctionnement, et prendre des décisions éclairées basées sur des données précises et actualisées.

3.2.4 Création de la base de données

Pour pouvoir créer la base de données Optiplant, il faut passer par différentes étapes :

- Étape de listage : le but est de lister toutes les entités comprises dans notre système ainsi que les champs qui les constituent.
- Étape de modélisation : les tables de la base sont représentées sous formes de schémas, les liaisons entre les entités sont définies, puis un deuxième schéma est fait pour montrer l'évolution des liens. (Voir MCD, MLD)
- Étape d'implantation sur un outil de gestion de base de données (PhpMyAdmin dans notre cas).

Finalement la BDD est constituée de 13 tables liées entre elles et ayant chacune un but précis. Cette architecture relationnelle soigneusement conçue permet d'éviter la redondance des données tout en garantissant leur intégrité et leur cohérence.

Chaque table répond à un besoin spécifique de gestion : tray (bacs de plantation), plants (plantes), group (groupes de plantes), alert (alertes), irrigation (arrosage), weather (météo), electric_production (production électrique), recipe (recettes de culture), shade_house (ombrières), sensor (capteurs), data (données des capteurs), picture (images), period (périodes de culture).

Maintenant que l'étape de listage des entités a été faite et que la création des 13 tables associées a été réalisée selon la modélisation suivante :

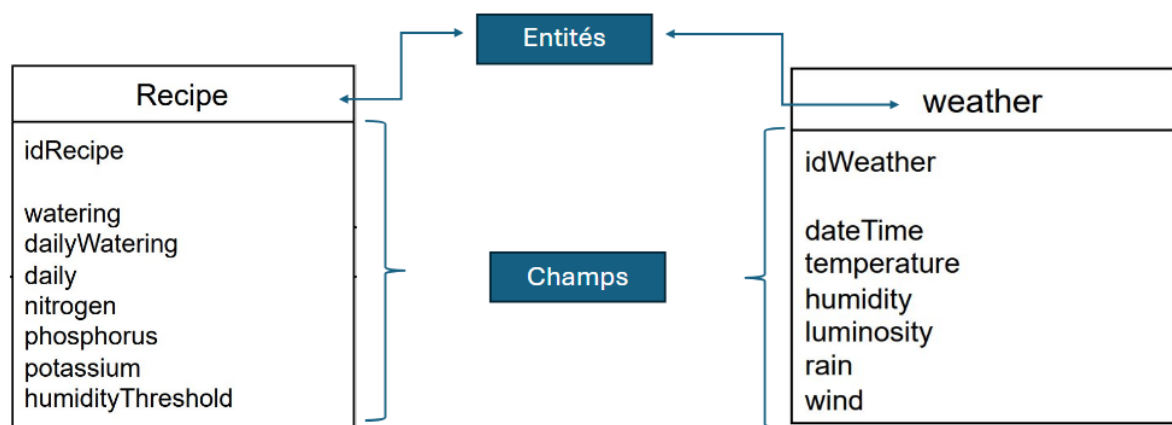


Figure 3 – Exemple de relation entre deux tables

Alors, il faut dès à présent passés à la modélisation MCD puis MLD.

3.2.5 MCD

Le Modèle Conceptuel des Données (MCD) de la base Optiplant permet de décrire la structure de la base de données en identifiant les entités principales du système ainsi que leurs relations.

Ce modèle met en évidence les interactions entre les différentes entités en illustrant les liens qui les unissent à travers des associations, exprimées par des verbes à l'infinitif. L'utilisation de ces verbes permet de définir clairement la nature des relations entre les entités.

Le MCD précise également les cardinalités, qui définissent le nombre minimum et maximum d'occurrences possibles entre les entités. Ces informations sont essentielles pour assurer la cohérence du modèle et anticiper les contraintes qui seront appliquées lors de la conception du Modèle Logique des Données (MLD).

En structurant visuellement l'organisation des données, le MCD constitue une étape clé pour préparer la transformation en tables relationnelles. Grâce à une définition rigoureuse des entités, des associations et des cardinalités, il facilite la mise en place d'une base de données cohérente et bien structurée.

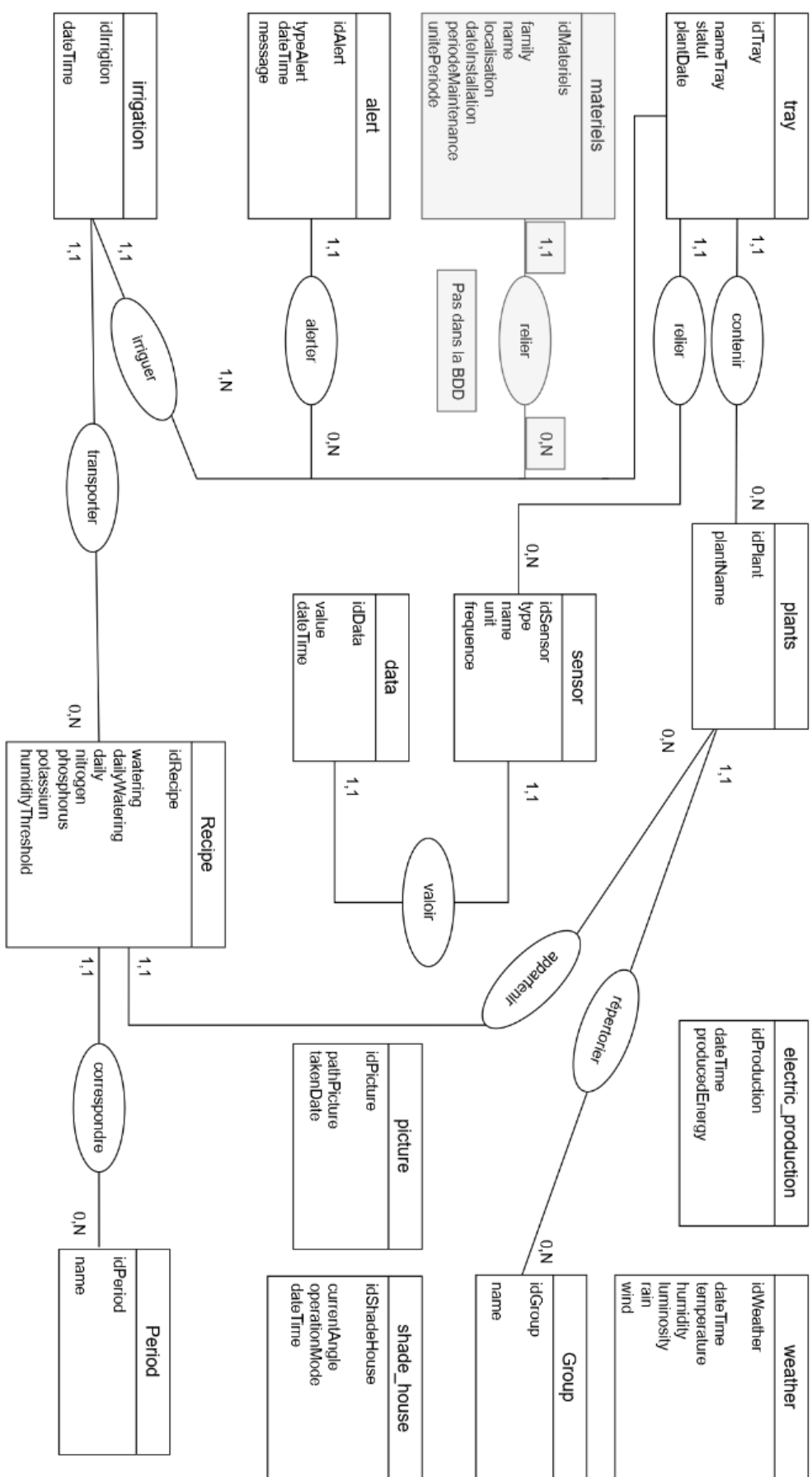


Figure 4 – MCD de la BDD

La table *tray* représente les bacs dans lesquels les plantes sont cultivées. Chaque bac peut contenir une plante, ce qui établit une relation de type 1 à 1 entre la table *tray* et la table *plant*. Chaque enregistrement dans la table *plant* est lié à un bac spécifique et appartient à un groupe, permettant ainsi de catégoriser les plantes en fonction de leurs types (Plantes à feuilles, Plantes à fruits, Herbes aromatiques, Légumes et Plante à fleurs). Une plante est associée à un groupe avec une relation de type 1 à 1, ce qui signifie que chaque plante appartient à un seul groupe. En revanche, chaque groupe peut contenir de 0 à N plantes, établissant une relation de type 0 à N entre les groupes et les plantes.

La table *irrigation* permet de suivre les événements d'arrosage des plantes. Un bac peut être lié à plusieurs événements d'irrigation, établissant une relation de type 1 à N entre *tray* et *irrigation*. De plus, un *bac* peut déclencher plusieurs *alertes* dans la table *alert* en cas de problème comme des niveaux d'humidité ou température anormaux, ce qui crée une relation de type 1 à N entre les bacs et les alertes.

La table *recipe* définit les besoins spécifiques des plantes pour favoriser leur pousse. Chaque recette est associée à une période et à une ou plusieurs plantes, créant ainsi une relation de type 1 à N entre *recipe* et *plant*, où chaque recette peut être liée à plusieurs plantes, mais chaque plante ne correspond qu'à une seule recette à la fois.

La table *sensor* enregistre des données environnementales comme la température, l'humidité ou la luminosité des bacs. Chaque capteur de la table *sensor* peut collecter plusieurs valeurs, créant une relation de type 1 à N entre *sensor* et *data*.

Enfin, la table *shade_house* gère l'ouverture et la fermeture des volets de l'ombrière en fonction des besoins des plantes. La table *materiels* est utilisée pour gérer la maintenance des capteurs et des actionneurs, mais elle ne sera pas incluse dans la base de données finale. Les données sur la production électrique sont également enregistrées dans la table *electric_production*, qui permet de suivre l'énergie produite par les équipements. Enfin, la table *weather* permet de collecter les données de la station météo KNX en fonction de temps.

Ce MCD sert de base pour la conception du MLD (Modèle Logique des Données) et donc de la structure de la base de données réelle.

3.2.6 MLD

Le Modèle Logique des Données de la base Optiplant est la traduction du Modèle Conceptuel des Données en tables relationnelles. Il permet de structurer la base en définissant les tables, leurs attributs et les relations entre elles.

Dans ce modèle, chaque entité du MCD est transformée en une table contenant les attributs correspondants. Les types de données sont précisés afin d'assurer la cohérence des informations stockées. Les relations entre les tables sont spécifiées à l'aide de clés étrangères, ce qui permet de maintenir l'intégrité des liens entre les données. Les clés primaires sont également identifiées afin de garantir l'unicité des enregistrements dans chaque table.

Ce modèle constitue une étape pour la préparation de l'implémentation SQL en établissant une structure précise et optimisée pour l'organisation des données.

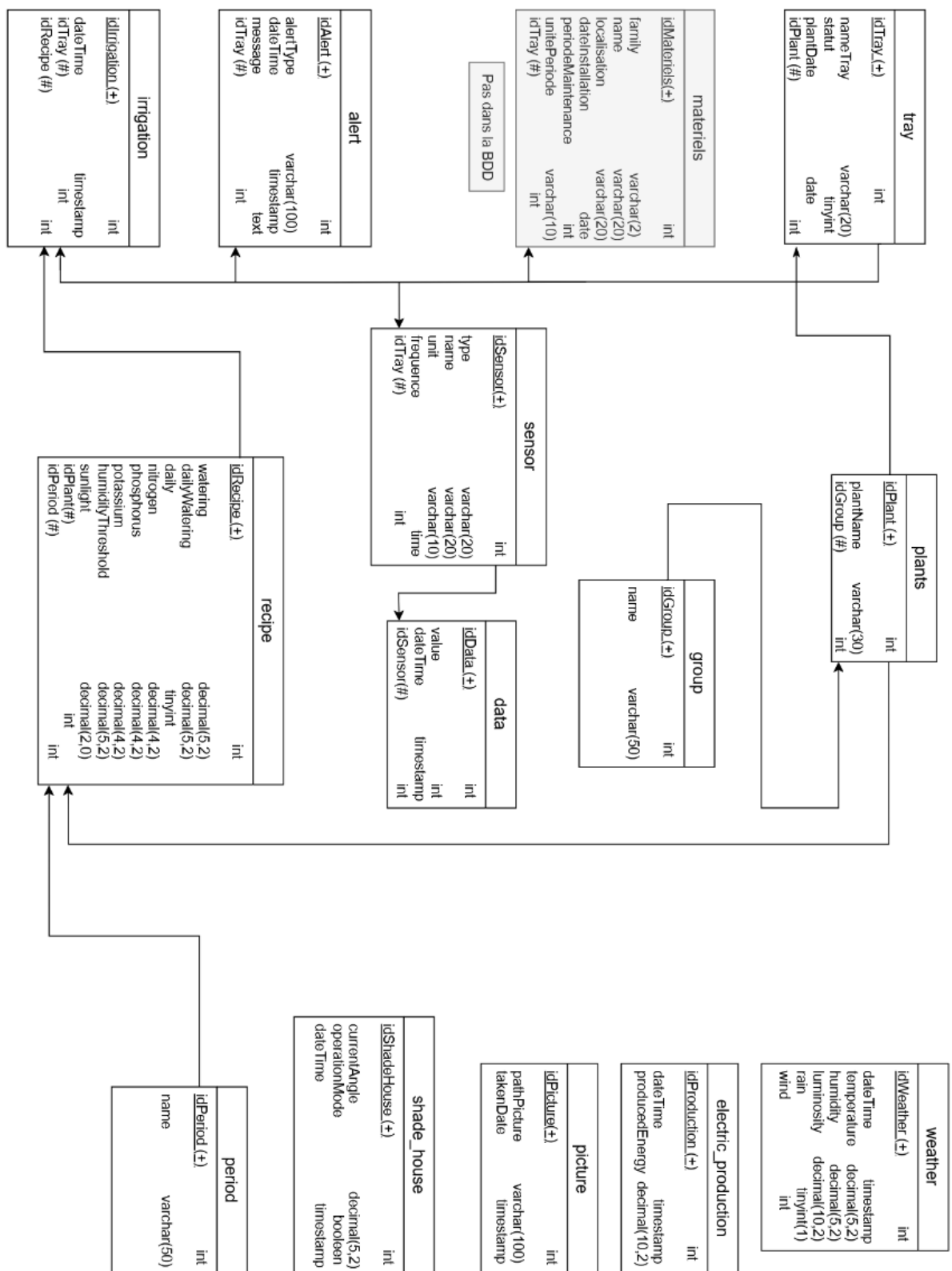


Figure 5 – MLD de la BDD

La table *tray* gère les bacs de plantation, chaque bac ayant un identifiant unique *idTray* et pouvant contenir une plante spécifique, référencée par *idPlant*. La table *plants* contient les informations sur les plantes, avec un identifiant unique *idPlant* et un lien vers leur groupe de culture dans la table *group*.

La gestion de l'irrigation se fait via la table *irrigation*, qui enregistre les événements d'arrosage. Chaque événement est lié à un bac par la clé étrangère *idTray* et à une recette spécifique par la clé *idRecipe*, définie dans la table *recipe*. Les recettes contiennent des informations sur la quantité d'eau, les nutriments et les conditions d'arrosage, et sont associées à une période de culture (semis, développement des racines, croissance végétative et floraison et fructification) définie dans la table *period*.

Les alertes relatives aux bacs sont stockées dans la table *alert*, où chaque alerte est liée à un bac. Les conditions climatiques sont enregistrées dans la table *weather*, qui collecte des données telles que la température, l'humidité et la luminosité.

Les capteurs installés dans les bacs sont référencés dans la table *sensor*, et les données mesurées par ces capteurs sont enregistrés dans la table *data*. La gestion de l'énergie produite par les installations est suivie dans la table *electric_production*, tandis que la table *shade_house* gère l'ouverture des volets de l'ombrière, avec des informations sur l'angle d'ouverture et le mode de fonctionnement.

Enfin, la table *materiels* non incluse dans la base de données finale permet de gérer les équipements et leur maintenance, et la table *picture* enregistre le chemin pour accéder aux images prises lors du Livestream dans l'ombrière.

Ce MLD permet une gestion complète et structurée de la culture, des conditions environnementales, de l'irrigation et des équipements associés.

C'est cette modélisation finale qui va être implantée dans l'outil de gestion phpMyAdmin.

4. Interface Web

4.1 Gestion des utilisateurs

Pour gérer les différents utilisateurs du site web, une base de données indépendante est créée. Celle-ci contient les différentes informations uniques à chaque utilisateur telles que sa dernière connexion ou son mot de passe.

4.1.1 Inscription sur le site

Pour se connecter au site internet, il faut obligatoirement posséder un compte. Pour cela, une page d'inscription est accessible. Dans celle-ci, l'utilisateur entre dans trois champs respectifs, son adresse e-mail, son nom d'utilisateur et son mot de passe. Pour les entrées d'e-mail et de mot de passe, les attributs `type="email"` et `type="password"` en HTML sont utilisés. Cela permet au navigateur de vérifier que dans l'entrée e-mail, un arobase et une extension sont bien présentes et de cacher la saisie du mot de passe en remplaçant chaque caractère par un point. Pour vérifier qu'un nom d'utilisateur ou un e-mail ne soit pas déjà utilisé, une requête SQL qui sélectionne les utilisateurs ayant le même email (et/ou nom d'utilisateur) est exécutée. S'il y en a, un message d'erreur s'affiche. Un message d'erreur s'affiche également lorsque le mot de

Le mot de passe ne respecte pas certaines contraintes. En effet, le code vérifie que la longueur du mot de passe est bien supérieure à 8 caractères, qu'il comporte bien une lettre minuscule, une lettre majuscule, un chiffre et un caractère spécial et qu'il n'y a aucun espace. Pour vérifier le respect de ces contraintes, une fonction d'expressions régulières (regex) est utilisée. La regex est une chaîne de caractère qui représente un modèle et qui permet de filtrer/gérer du texte. Dans cette application, la fonction vérifie la correspondance entre le mot de passe et des expressions – une lettre majuscule par exemple – ou l'absence d'espace. Lorsque tous les remparts de vérification sont franchis, le mot de passe est haché avec la fonction `password_hash` de PHP qui applique un algorithme de hachage fort et irréversible. Avant de valider l'inscription, l'utilisateur peut sélectionner un avatar en photo de profil. Cette fonctionnalité a été ajoutée à des fins esthétiques et ludiques, mais permettra aussi aux administrateurs d'identifier rapidement les différents membres. Une fois l'inscription validée, une requête insère une nouvelle ligne dans la table Users en enregistrant le nom, l'e-mail, la photo de profil et le mot de passe haché. La session est alors enregistrée dans la table.

Inscription

Ce nom d'utilisateur est déjà utilisé.

Adresse e-mail:

Nom d'utilisateur:

Mot de passe:

Choisir la photo de profil



S'inscrire

← Page de connexion

Figure 6 – Page d'inscription

4.1.2 Connexion et déconnexion

La connexion se fait dans un premier temps en vérifiant dans la table que pour le nom d'utilisateur entré, le mot de passe correspond grâce à la fonction `password_verify` en PHP. Une session est alors activée. La déconnexion se fait quant à elle grâce à un code PHP accessible par l'appui sur un bouton déconnexion sur le site. Celle-ci supprime tous les cookies et les variables de session avant de la détruire avec la fonction `session_destroy()` et de renvoyer vers la page de connexion.

4.1.3 Page de profil

Lorsque l'utilisateur est connecté sur le site, il a accès à une page pour gérer son profil. Différentes sections composent cette page. Il pourra changer son mot de passe, sa photo de profil, mais également se déconnecter. Il pourra aussi choisir son mode d'affichage et celui-ci sera enregistré dans la table afin d'être conservé lors des prochaines connexions. Chaque utilisateur a un rôle, soit membre (par défaut), soit administrateur si un admin le promeut. Les administrateurs ont accès à une section privée dans laquelle ils peuvent justement promouvoir les autres membres, mais également consulter différentes informations notamment l'heure de la dernière connexion. Cela peut s'avérer utile pour contrôler les actions des consultants du site. Les traces pourront être retrouvées en cas de sabotage par exemple. En figure 7 se trouve une image de la section réservée aux admins.




Photo de profil	Nom d'utilisateur	Dernière connexion	Rôle	Action
	othmane83	2025-03-23 22:48:12	admin	
	test	2025-03-06 12:11:03	membre	

Figure 7 – Section réservée aux administrateurs

4.1.4 Base de données utilisateurs

En Figure 8 se trouve la table de la base de données utilisateurs.

id	username	password	email	lerole	profile_photo	last_login	mode
16	othmane83	\$2y\$10\$EJnHbbD/ZT/uW90cD21U3OBiRN525n5CA.7JcEFjXil...	elbertalothmane0@gmail.com	admin	images/nyquit3.jpg	2025-03-16 22:58:58	default
15	test	\$2y\$10\$rTLN.oOnqkKlgLHRr816SetaBNRntLjEiRn6H.A.GO0...	test@test.fr	membre	images/nyquit1.jpg	2025-03-06 12:11:03	default

Figure 8 – Table de la base de données utilisateurs

4.2 Interactions avec les bacs

La fonctionnalité première du tableau de bord est de recenser toutes les informations les plus importantes et qu'elles soient facilement repérables.

Il y a tout d'abord, l'affichage des données météo qui proviennent de openweather via un script JavaScript, mais une fonctionnalité a été ajoutée afin d'y voir l'heure de mise à jour et elle provient d'une requête SQL qui permet de s'assurer qu'elle sera toujours correcte.

Puis l'affichage des alertes et enfin ce qui va nous intéresser ici, la visualisation de la contenance des bacs. Ces dits bacs peuvent être visualisés indépendamment à l'aide d'un bouton. Chaque bac a sa propre page afin d'y voir les données nécessaires aux plantes. Ces données sont prises depuis la base de données à l'aide de requête SQL.

Lorsqu'un clic sur l'un des bacs est effectué, une requête va donc être lancée à la base de données avec le numéro du bac qui y est associé. Une fois ceci fait, la page du bac s'affiche et est présentée avec trois onglets : sommaire, photos et données. Chaque onglet est vu comme suit.

L'onglet photos permet de voir le bac avec des photos prise par la caméra.

L'onglet « données » permet quant à lui de pouvoir visualiser les données d'irrigations et des capteurs ainsi que les dernières alertes.

Et enfin, l'onglet sommaire a pour objectif de faire directement comprendre à l'utilisateur ce que contient le bac, à quelle période la plante se trouve et les conditions d'humidité et de température que la plante est en train de subir. Ceci est possible grâce à des graphes faits en JS qui permet, au-delà de l'aspect visuel, de prévenir et d'ajouter dans la base de données une alerte si les conditions n'atteignent pas un certain seuil ou au contraire les dépassent.

4.2.1 Amélioration possible

Une amélioration possible est d'ajouter un graphe indiquant le temps d'ensoleillement qu'a eu la plante afin de faire varier la position des ombrières. Cela n'a pas été fait, car cette donnée n'était pas à notre disposition.

4.3 Interactions avec les recettes

L'un des principaux objectifs du projet était de permettre aux utilisateurs d'interagir facilement avec les recettes stockées dans la base de données. Cela inclut l'affichage, l'ajout, la modification et la suppression de recettes directement via l'interface web.

Pour cela, une page dédiée aux recettes a été créée, permettant d'interagir avec la table "recipes" de la base de données. Les utilisateurs peuvent y accéder de deux manières : via l'onglet "Recettes" dans le leader de navigation ou via le bouton "+ d'infos" dans la section "Les Recettes" du tableau de bord. Une fois sur cette page, ils disposent d'un header pour naviguer entre les différentes sections du site et d'un bouton "Retour au tableau de bord" pour revenir facilement au Dashboard.

Les recettes sont affichées sous forme de liste comprenant pour chacune un numéro d'identification, la plante associée ainsi que trois boutons permettant de voir les détails, de modifier ou de supprimer la recette. Un bouton "Ajouter" est également disponible en bas de page pour permettre la création de nouvelles recettes. L'affichage de ces recettes repose sur une requête SQL qui récupère les entrées de la table recipes. Ces données sont ensuite affichées dynamiquement dans une table HTML grâce à une boucle PHP.

Le bouton "Voir détails" ouvre un modal Bootstrap affichant les informations détaillées d'une recette sous forme de tableau. Ces informations sont récupérées via une requête SQL qui joint les tables "recipes, periods, plants et irrigation" et mises à jour dynamiquement en JavaScript lorsqu'un utilisateur clique sur le bouton.

Le bouton "Modifier" ouvre un modal prérempli avec les informations actuelles de la recette. Les champs numériques (comme le seuil d'humidité ou l'arrosage) sont modifiables via des champs de texte, tandis que les champs avec des options prédéfinies (comme la période et la plante) sont gérés via des menus déroulants dynamiquement remplis depuis la base de données. Une fois les modifications effectuées, l'utilisateur peut choisir d'annuler l'action ou de l'enregistrer. Lorsque l'utilisateur clique sur "Enregistrer", une requête SQL de type UPDATE est exécutée dans le fichier update-recipe.php qui met à jour la recette. Si l'utilisateur annule, le modal se ferme sans sauvegarder les modifications.

Le bouton "Supprimer" permet de supprimer une recette de la base de données. Pour éviter les erreurs de manipulation, un message de confirmation est affiché avant la suppression. Si l'utilisateur confirme, une requête SQL DELETE est exécutée via le fichier delete-recipe.php pour supprimer la recette correspondante.

Le bouton "Ajouter" ouvre un modal similaire à celui de modification, mais vide. L'utilisateur saisit les informations nécessaires, qui sont ensuite insérées dans la base de données si la requête est validée. Le fichier update-recipe.php distingue une modification, d'une insertion en vérifiant la présence ou non d'un identifiant de recette.

Pour améliorer l'expérience utilisateur, un message de confirmation s'affiche lorsque l'ajout, la modification ou la suppression d'une recette est réussie. Ce message est généré en passant un paramètre success=1 dans l'URL après l'exécution de la requête. Dans recettes.php, le code détecte ce paramètre et affiche un message de confirmation en vert et disparaît automatiquement après trois secondes grâce à un code JavaScript.

4.3.1 Amélioration possible

Bien que les objectifs principaux aient été atteints, certaines améliorations pourraient être apportées. Par exemple, les messages de confirmation pourraient être personnalisés en fonction de l'action effectuée (ajout, modification ou suppression) et indiquer la recette concernée. De plus, une gestion des rôles utilisateurs pourrait être implémentée pour réserver les fonctionnalités de modification, suppression et ajout aux administrateurs, tandis que les utilisateurs standards ne pourraient que consulter les détails des recettes.

5. CSS

5.1 Design du site web

La première étape avant de modifier le site web ou les fichiers CSS consiste à établir une liste des critères attendus, incluant le thème de couleurs, le fonctionnement, la décoration. Ce projet étant celui d'une ombrière automatisée, l'utilisateur doit comprendre l'idée du projet dès son arrivée sur le site. Il est important de garantir l'accessibilité en veillant à ce que les couleurs, décorations et la disposition des éléments n'impactent pas l'expérience utilisateur.

Le site web attendu se caractérise par :

- Un thème de couleurs rappelant la nature, l'extérieur, les plantes.
- Des éléments de décoration liés au thème, mais en nombre limité.
- Une organisation simple et moderne.
- Un design responsive.

Le site repose sur le framework Bootstrap, qui facilite la création d'une structure responsive cohérente, mais limite la personnalisation. Pour adapter l'identité visuelle du projet, certains éléments fait avec Bootstrap ont été remplacés progressivement, notamment le header, entièrement refait sur mesure.

Une des premières idées pour le site web a été l'insertion d'animations de plantes, pour rappeler la culture dans les bacs. Grâce au site lottiefiles.com, il a été possible de trouver une animation de plantes en pleine croissance. Son insertion dans le site web a permis de guider le choix de couleurs et du design global.

Comme sur la plupart des sites, un header et un footer ont été intégrés. Le choix a donc été fait de mettre un footer marron, pour rappeler la terre, et d'y ajouter l'animation plante qui pousse comme décoration discrète. Quant au header, celui-ci est de couleur bleue, pour rappeler le ciel.

Il est possible de circuler facilement sur les différentes pages grâce aux boutons ou liens disponibles dans le header. Ici, il y a 4 pages principales : Dashboard, Recettes, Livestream et Profil. Avec la création de cette dernière, celle-ci a donné l'idée de rajouter comme sur beaucoup de sites, une photo de profil en haut à droite. Pour que l'utilisateur comprenne la logique de terre et de ciel, le texte dans le header a été mis en blanc (nuage) et pour plus de clarté, le lien de la page active est souligné. Une dernière animation a été ajoutée à gauche dans le header, celle d'un soleil (provenant de lottiefiles.com).

L'intégration des animations est simplifiée grâce au site de LottieFiles, qui fournit un code à insérer directement, permettant à tout le groupe de visualiser l'animation prise en ligne, sans avoir à l'héberger localement.

Enfin, il est important de mettre le nom du site web ou plutôt ici l'intitulé du projet soit « SAE Ombrière ». Comme précisé précédemment, un bouton accessibilité est présent en haut du site web, celui-ci n'est certes pas très esthétique, mais il reste visible par tous.

Comme chaque partie du site web a été faite en parallèle notamment le contenu du site (les bacs, recettes, Livestream...), ces pages n'ont pas été particulièrement personnalisées, mais un thème global a été mis en place : un fond blanc, une police d'écriture noire, et des boutons verts pour rappeler les plantes. Le header devant se trouver sur chaque page du site, pour faciliter son intégration, un fichier individuel nommé header.php a été créé et est inséré dans chaque fichier avec :

```
<?php include("header.php"); ?>
```

De cette manière, un seul fichier est nécessaire si l'on souhaite faire des modifications du header.

Une fois tout mis en place, le rendu est le suivant :

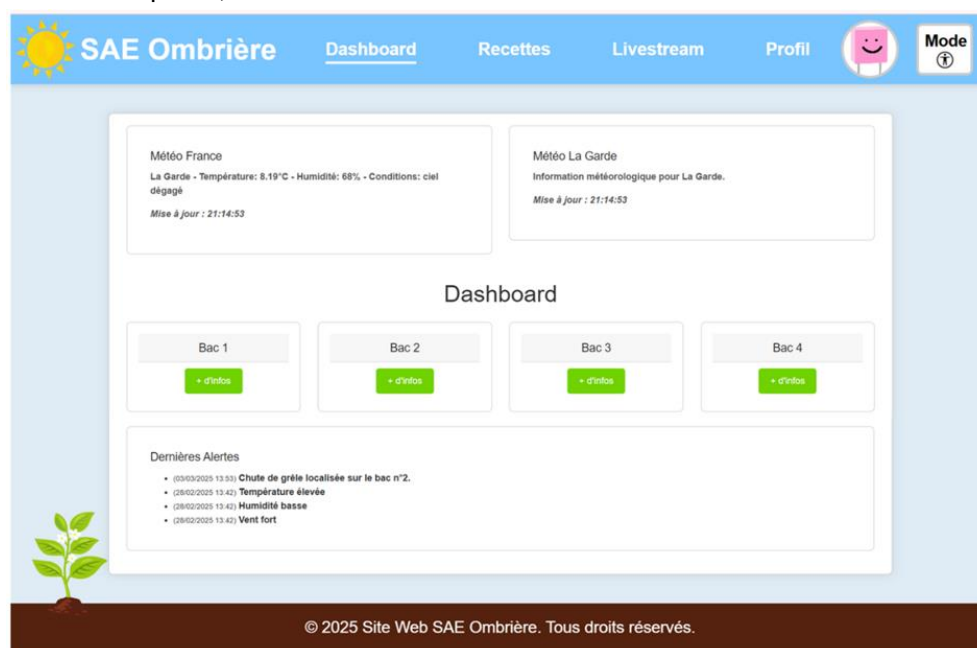


Figure 9 – Rendu Final du design du Site Web

5.2 Esthétique et Accessibilité

5.2.1 RGAA

Le RGAA (Référentiel Général d'Amélioration de l'Accessibilité) est un ensemble de 106 normes destinées à assurer l'accessibilité des sites web français. Ces normes varient d'un pays à l'autre et sont définies selon différents critères et niveaux d'exigence. Dans le cadre du RGAA, ces niveaux sont classés en trois catégories : A, AA et AAA, en fonction du degré de conformité requis.

Une autre norme française en matière d'accessibilité est AccessiWeb, mais pour ce projet, seules les exigences du RGAA ont été prises en compte.

L'une des premières options développées est la possibilité de changer le style de couleur, permettant ainsi de commencer à travailler sans avoir à attendre que le site soit entièrement fini.

L'objectif de cette fonctionnalité est de permettre aux utilisateurs de choisir parmi plusieurs modes d'affichage :

- Mode par défaut
- Mode sombre (darkside)
- Mode contraste élevé (contrasté)
- Mode deuteranope
- Mode protanope
- Mode tritanope
- Mode achromatope

Chaque mode de couleur possède son propre fichier CSS. Lorsqu'un mode est sélectionné, le fichier CSS correspondant est appliqué dynamiquement. Le mode sélectionné est sauvegardé dans la base de données et associé au compte de l'utilisateur. Ainsi, en cas de rechargement ou de reconnexion, le site conserve automatiquement le dernier style CSS choisi.

Cette fonctionnalité est directement liée aux exigences du RGAA, permettant une adaptation du site à différents handicaps, comme le daltonisme.

La base CSS du site web étant créée, il faut s'assurer que celui-ci est accessible et responsive.

Même si la page web telle qu'elle est rendue responsive, le header contenant plusieurs informations, celles-ci risquent de se mélanger et de ne plus être accessible par l'utilisateur. À partir de la taille des éléments, il a été défini que si la largeur de la page est inférieure à 1300 px, le header changera d'organisation. Pour cela une nouvelle partie est ajoutée dans les fichiers CSS:

```
@media (max-width: 1300 px) {...}
```

Les lignes présentes dans @media permettent de cacher certains éléments comme les liens ou la photo de profil. À taille minimale, il ne reste donc que l'animation soleil, le titre tout de même rétrécis, le bouton accessibilité et un nouveau bouton, celui du menu.

Dans le fichier header.php, un bouton menu a été créé et celui-ci ne s'affiche que lors du changement de taille de la page. Lorsqu'il est cliqué, le menu affiche les 4 liens vers les pages Dashboard, Recettes, Livestream et Profil. Comme il est possible de le voir ci-dessus, un détail a été changé : le soleil est remplacé par une lune dans le mode nuit.



Figure 10 – Header en mode Responsive

Dans l'idéal, un site qui respecte à 100 % le RGAA, serait très épuré, voire trop épuré. Aucune décoration inutile ne devrait être présente, les couleurs choisies devraient convenir à tous les types de problème visuel..., ce qui est compliqué pour un projet ayant une certaine direction artistique.

Les animations par exemple, peuvent être une distraction pour certaines personnes. Cependant, le soleil en rotation ou la plante qui pousse, apportent du dynamisme. Ces choix esthétiques ne sont pas seulement décoratifs : ils servent aussi à immerger l'utilisateur dans l'univers du projet et à rendre l'expérience plus engageante.

Il est donc important de prendre en compte le fait que l'accessibilité passe aussi par la décoration : pour les personnes atteintes d'achromatopsie (vision en noir et blanc) ou de daltonisme, la distinction entre les éléments ne doit pas uniquement se faire par la couleur, mais par les formes, le mouvement ou les images.

Pour respecter au maximum le règlement sur l'accessibilité, les animations sont donc gardées, mais peuvent être arrêtées si elles sont cliquées. Pour faciliter l'utilisation d'un assistant technologique, pratiquement tous les éléments du site contiennent un texte alternatif, qui décrit l'élément, notamment si celui-ci ne s'affiche pas. Pour les animations et décorations, l'ajout de `aria-hidden="true"`, permet de masquer les décorations aux lecteurs d'écran afin d'éviter toute confusion. Enfin, le site web est compatible avec une navigation intuitive puisque tous les éléments interactifs sont accessibles via TAB et ENTER, facilitant l'accès aux utilisateurs naviguant au clavier.

Pour ce qui est du CSS des différents modes visuels, les couleurs choisies, notamment pour le daltonisme, n'ont pas été particulièrement modifiées. D'après les exemples d'internet, les personnes atteintes de protanopie ou de deutéranopie peuvent tout de même voir la couleur jaune et bleu donc le header reste bien contrasté. Pour ceux atteints de tritanopie, le ciel reste bleu (légèrement turquoise) mais le soleil devient rose clair/blanc, soit toujours un bon contraste. Le footer, lui, n'est pas changé puisqu'il n'est pas le plus important et est assez foncé pour être contrasté. Cependant, les couleurs des boutons sont changées puisque la plupart des types de daltonisme ne voient pas ou mal le vert. En fonction, ceux-ci sont remplacés par du bleu et du jaune ou du bleu et du rose.

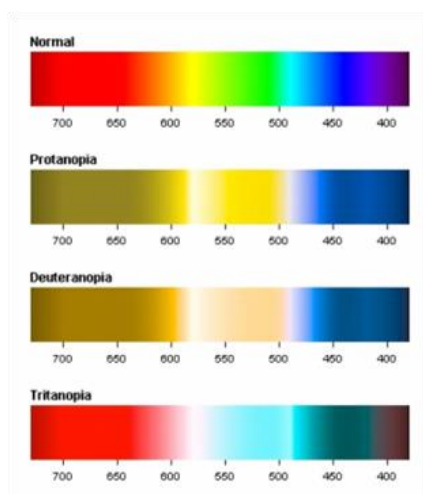


Figure 11 – Perception des couleurs sous différents types de daltonisme

Le cahier des charges du projet imposait le respect d'au moins 10 normes parmi les 106 du RGAA. Afin de sélectionner les normes qui seraient respectées ou non, une lecture de celles-ci a été réalisée puis elles ont été regroupées en 42 normes “simplifiées” dans un fichier Word, certaines étant redondantes ou très similaires.

Un tableau Excel a ensuite été utilisé pour trier et sélectionner les normes à appliquer. Certaines normes ont été respectées automatiquement puisqu'elles concernent directement la structure du code HTML. Un des défis rencontrés a été que la vérification RGAA devait être effectuée sur le site fini, ce qui a placé cette tâche en dernière étape soit après que le reste de l'équipe du site a terminé.

Pour vérifier la conformité aux normes choisies, une approche en plusieurs étapes a été adoptée:

1. Lecture et analyse du code HTML afin d'effectuer une première validation.
2. Utilisation de l'extension de navigateur Wave pour identifier d'éventuelles erreurs supplémentaires.

Wave est une extension qui analyse le code HTML d'une page et détecte les problèmes d'accessibilité en fonction des normes internationales. Cette extension étant américaine, elle ne se base pas directement sur le RGAA français. Cependant, les principes d'accessibilité étant globalement similaires, les corrections et suggestions proposées par Wave se sont avérées relativement cohérentes avec plusieurs normes sélectionnées.

Cette méthodologie a permis de réaliser rapidement la vérification du site afin qu'il respecte un minimum de 10 normes RGAA tout en garantissant une bonne accessibilité du site.

Bien que le site soit déjà fonctionnel et accessible, plusieurs améliorations peuvent encore être apportées comme des nouveaux modes (dyslexique, zoomer, minimaliste ou personnalisable comme le site web de l'ENSEA). Il faudrait également rendre les boutons adaptés aux modes (changer les boutons Bootstrap). Chaque mode d'accessibilité devrait avoir un bouton visuellement identifiable sans nécessité de lecture. Enfin, il faudrait surtout continuer la personnalisation : chaque page du site devrait avoir un design particulier pour renforcer l'identité visuelle du projet.

Avec ces améliorations, le site pourrait offrir une expérience encore plus inclusive, ergonomique et esthétiquement aboutie.

6. API Météo

6.1 Contexte et objectif

Cette section traite de l'implémentation de la récupération des données météo sur une API distante. Elle permet d'obtenir des données pour corroborer celles fournies par la station du système ou le cas échéant de prendre le relai en cas de panne. Par ailleurs, l'utilisation d'un API externe permet d'obtenir des prévisions à plus ou moins long terme qui pourraient être utilisées pour prévoir certaines actions.

Pour ce faire, l'API OpenWeather a été implémentée pour permettre de récupérer et de visualiser des informations clés telles que la température, l'humidité, les conditions météorologiques, ainsi

que des données supplémentaires comme la vitesse du vent, la couverture nuageuse, et les heures de lever et coucher du soleil.

6.2 Fonctionnalités et structure

Cette fonctionnalité repose sur un fichier JavaScript (`script.js`) qui gère la logique de récupération des données via l'API OpenWeatherMap. Les informations sont ensuite injectées dynamiquement dans une structure HTML prévue à cet effet. L'interface affiche des éléments tels que le nom de la ville, la température, l'humidité et une description des conditions météorologiques. Le code est conçu pour être facilement modifiable, permettant aux utilisateurs de changer la ville cible ou d'ajuster les unités de mesure (Celsius ou Fahrenheit) en modifiant quelques paramètres.

6.3 Installation et utilisation

Pour utiliser le code créé, il suffit de télécharger ou de cloner le projet "API-Meteo-France" (qui porte le mauvais nom), puis d'ouvrir le fichier HTML dans un navigateur web. Une connexion Internet est nécessaire pour interroger l'API. Toutefois, cette connexion ne pose pas de problème même s'il y a un proxy, car la requête est envoyée côté client. La clé API OpenWeatherMap est intégrée dans le code, mais elle peut être remplacée par une clé personnelle pour une utilisation personnalisée. L'affichage des données est automatique au chargement de la page, mais une fonction `getWeather()` permet de rafraîchir manuellement les informations.

6.4 Données météorologiques et traitement

Les données retournées par l'API sont structurées en JSON et incluent des informations détaillées sur la météo. Parmi les champs les plus utiles, on trouve :

- **Température** (`data.main.temp`) : La température actuelle
- **Humidité** (`data.main.humidity`) : Le taux d'humidité (en %)
- **Conditions météorologiques** (`data.weather[0].description`) : Une description textuelle des conditions (par exemple, "ciel dégagé" ou "pluie légère")
- **Vent** (`data.wind.speed`, `data.wind.deg`) : La vitesse du vent (en m/s) et sa direction (en °)
- **Couverture nuageuse** (`data.clouds.all`) : Le pourcentage de couverture nuageuse
- **Lever et coucher du Soleil** (`data.sys.sunrise`, `data.sys.sunset`) : Les heures de lever et coucher du soleil en timestamp Unix.

6.5 Conclusion

Cette fonctionnalité constitue une solution efficace et modulable pour visualiser les données météorologiques en temps réel, permise par son architecture simple et à sa flexibilité. L'utilisation de l'API OpenWeatherMap garantit des données précises et à jour et l'acquisition de l'ensemble des données permet une grande modularité dans l'affichage.

7. Livestream

Ce système de surveillance vidéo en direct a été conçu pour permettre une supervision visuelle à distance, bien que pour l'instant, la diffusion soit limitée au réseau local, ce qui nécessite d'être physiquement présent à l'université pour y accéder.

Le système repose sur l'utilisation de FFmpeg, un outil de traitement vidéo, pour capturer et diffuser en direct la vidéo provenant d'un périphérique vidéo, comme une caméra. Le flux vidéo

est enregistré localement et peut être consulté en temps réel via un serveur HTTP, accessible depuis un navigateur web. La diffusion en direct est basée sur le protocole HLS (HTTP Live Streaming), couramment utilisé pour le streaming vidéo. Le flux vidéo est découpé en segments courts au format .ts (Transport Stream), qui sont enregistrés en boucle avec un nombre limité de fichiers (ici, trois fichiers). Lorsqu'un nouveau segment est enregistré, le plus ancien est écrasé, garantissant que seuls les segments les plus récents sont conservés. Cette méthode optimise l'utilisation du stockage tout en maintenant un flux vidéo continu. Le fichier .m3u8, quant à lui, est un fichier de playlist utilisé par le protocole HLS. Il contient la liste des segments vidéo .ts disponibles pour la lecture et est mis à jour dynamiquement pour refléter les segments vidéo les plus récents, assurant une lecture fluide et continue du flux vidéo. Cette méthode offre une latence réduite, le flux vidéo étant presque en temps réel, avec un léger délai dû à la durée des segments (environ 10 secondes dans notre cas). Enfin, elle est compatible avec les navigateurs modernes et les appareils mobiles, facilitant l'accès au flux vidéo depuis différents dispositifs. En parallèle de la diffusion en direct de vidéo, le système capture des images à intervalles réguliers, intervalles définis dans un fichier de configuration permettant de les modifier facilement. Ces images serviront par la suite à créer des time-lapses qui montrent l'évolution des plantes sous l'ombrière.

En pratique, les utilisateurs peuvent accéder au flux vidéo en direct via un navigateur web en se connectant au serveur HTTP, à condition d'être sur le même réseau que le serveur puisque ce dernier n'est pas déployé. Cela permet de surveiller l'ombrière en temps réel et de réagir rapidement en cas de problème, comme des dégâts visibles ou des anomalies. Les images capturées à intervalles réguliers sont sauvegardées dans un répertoire spécifique, et leur chemin d'accès est envoyé dans une base de données à l'aide de requêtes SQL. L'envoi des chemins d'accès dans la base de données permet de vérifier rapidement le bon fonctionnement du script de capture d'image, sans avoir besoin de se connecter directement à la machine hébergeant le serveur. Étant donné que les noms des images capturées incluent la date et l'heure de capture, la présence de ces chemins d'accès dans la base de données permet de confirmer que les intervalles de capture respectent bien les paramètres définis. Cela offre un moyen simple et efficace de valider le processus de capture à distance.

Le fonctionnement du système repose sur plusieurs composants clés. Le script Python (*server.py*) orchestre l'ensemble du processus. Le choix d'utiliser un script Python plutôt qu'un script PHP pour la création du Livestream est principalement motivé par la gestion du CORS (Cross-Origin Resource Sharing). L'activation de CORS sur un serveur est essentielle pour autoriser des ressources d'un site web (comme des scripts) à être demandées depuis un domaine différent de celui d'où le site est servi. Bien que plusieurs solutions techniques existaient pour implémenter le CORS, Python a été privilégié en raison de sa simplicité de mise en œuvre côté serveur. Lors du lancement du script Python, la configuration du système est chargée à partir d'un fichier JSON (*config.json*), qui contient des paramètres tels que le périphérique vidéo (*/dev/video0*), le format vidéo (640x480), les répertoires de sortie (*stream* et *images*), et les intervalles de capture d'images (*image_period*). Le script Python démarre également les scripts Bash (*stream.sh* et *acquireimg.sh*) pour gérer respectivement le streaming vidéo et la capture d'images. Le script *stream.sh* utilise FFmpeg pour diffuser en direct la vidéo via le protocole HLS, tandis que *acquireimg.sh* capture les images à partir des segments vidéo .ts et les sauvegarde dans un répertoire spécifique. Enfin, le script Python envoie le chemin de ces images dans la base de données via des commandes SQL pour un suivi structuré.

Le système est configuré pour fonctionner sur des systèmes Linux, comme en témoigne l'utilisation de scripts Bash (stream.sh, acquireimg.sh, et dimw.sh) mais a aussi été porté sur Windows en mettant ces scripts au format Bat. Lors du premier lancement du serveur sur le LattePanda, aucune vidéo n'était enregistrée. Ce problème était lié à l'utilisation de FFmpeg, qui ne parvenait pas à lancer l'enregistrement. Initialement, l'hypothèse retenue pointait une incompatibilité entre le LattePanda et la caméra utilisée pour les tests. Cependant, en utilisant la commande `ffplay`, la diffusion de la caméra fonctionnait correctement, écartant cette première piste. Une seconde hypothèse a alors été envisagée : le problème pourrait provenir des options utilisées dans la commande FFmpeg. Après plusieurs tests, une solution a été trouvée : pour la première utilisation, il faut créer à l'aide de FFmpeg un fichier .mp4 initial. Une fois cette étape réalisée, la commande pour l'enregistrement de la vidéo pour le Livestream fonctionne sans problème.

Pour garantir un fonctionnement constant sans intervention manuelle, le script `dimw.sh` a été développé. Ce script résout un problème lié à l'enregistrement en capturant une très courte séquence vidéo au format .mp4, puis en la supprimant immédiatement. Cette opération permet de s'assurer que le périphérique vidéo est correctement initialisé avant de démarrer le streaming, évitant ainsi tout dysfonctionnement ultérieur.

8. Mise en service LattePanda

La mise en service du système sur la LattePanda a été une étape cruciale pour garantir le bon fonctionnement de l'ensemble des composants, incluant la base de données MySQL, le serveur web, l'API météo, et le Livestream. Cette section décrit les étapes clés de cette mise en service, en mettant l'accent sur l'ouverture des ports, l'installation des logiciels nécessaires, la configuration de phpMyAdmin, et l'accès distant à la base de données.

8.1 Ouverture des ports

Pour permettre au système de communiquer avec les autres appareils du réseau, plusieurs ports ont dû être ouverts sur la LattePanda. Le port 3306 a été ouvert pour MySQL, permettant ainsi aux autres ordinateurs du réseau de se connecter à la base de données. Le port 80 a été ouvert pour le serveur web, hébergeant l'interface utilisateur, phpMyAdmin, et l'API météo. Enfin, le port 20 a été ouvert pour des besoins spécifiques, tels que les transferts de fichiers via FTP. Ces ports ont été configurés à l'aide du pare-feu `ufw` sous Linux, avec des commandes spécifiques pour autoriser les connexions entrantes sur ces ports.

```
sudo ufw allow 3306/tcp # Autoriser le port 3306 pour MySQL
```

8.2 Installation des logiciels nécessaires

Plusieurs logiciels ont été installés sur la LattePanda pour assurer le bon fonctionnement du système. MySQL a été installé pour héberger les bases de données « Optiplant » et « Comptes Utilisateurs », tandis que phpMyAdmin a été configuré pour offrir une interface graphique de gestion de la base de données. Apache a été installé pour servir de serveur web, hébergeant l'interface utilisateur et l'API météo. Python a été utilisé pour exécuter les scripts backend, y compris le serveur Flask pour le Livestream. FFmpeg a été installé pour la capture et le streaming vidéo, et Git a été utilisé pour cloner et mettre à jour les dépôts de code. Ces installations ont été réalisées via des commandes simples sous Linux, garantissant une configuration rapide et efficace.

```
sudo apt install mysql-server phpmyadmin apache2 python3 python3-pip ffmpeg  
git
```

8.3 Configuration de phpMyAdmin

Une fois phpMyAdmin installé, plusieurs étapes de configuration ont été nécessaires pour le rendre accessible et sécurisé. phpMyAdmin a été configuré pour être accessible via le navigateur à l'adresse `http://192.168.XXX.XXX/phpmyadmin`. Un utilisateur dédié a été créé dans MySQL pour phpMyAdmin, avec des permissions restreintes pour éviter les accès non autorisés. Le fichier de configuration d'Apache a également été modifié pour permettre l'accès à phpMyAdmin depuis le réseau local. Ces modifications ont permis de garantir une interface de gestion de la base de données à la fois accessible et sécurisée.

8.4 Configuration de MySQL pour l'accès distant

Pour permettre à d'autres ordinateurs du réseau d'accéder à la base de données MySQL, plusieurs configurations ont été appliquées. Le fichier de configuration MySQL (`/etc/mysql/mysql.conf.d/mysqld.cnf`) a été modifié pour écouter sur toutes les interfaces réseau, et non seulement sur `localhost`.

```
bind-address = 0.0.0.0
```

Un utilisateur MySQL avec des permissions d'accès distant a été créé, permettant aux autres ordinateurs du réseau de se connecter à la base de données. Ces modifications ont été suivies d'un redémarrage du service MySQL pour appliquer les changements. Cette configuration a permis une collaboration efficace entre les différents utilisateurs du réseau.

```
CREATE USER 'remote_user'@'adresse_ip_pc' IDENTIFIED BY 'password';  
GRANT ALL PRIVILEGES ON Optiplant.* TO 'remote_user'@'adresse_ip_pc';  
FLUSH PRIVILEGES ;
```

8.5 Accès à la LattePanda depuis le réseau local

Pour permettre aux autres ordinateurs du réseau d'accéder à la LattePanda, l'adresse IP de la LattePanda a été partagée avec les utilisateurs. Les utilisateurs peuvent accéder à l'interface utilisateur, à phpMyAdmin, et au Livestream via cette adresse IP. Cette configuration a permis une utilisation collaborative du système, avec un accès centralisé à tous les services.

Il est à noter que pendant la phase d'installation du LattePanda, il a fallu utiliser un autre réseau que le réseau filaire de l'université. En effet, le proxy n'est pas configuré pour réaliser ce genre de tâche. De plus, il n'est pas envisageable de se connecter avec un compte personnel sur le proxy, pour des raisons évidentes de protection des données.

Cependant, dans le cadre d'une utilisation normale, il est possible de connecter le server au réseau filaire de l'université dans la mesure où le serveur n'effectue pas de requête sortante, se contentant de répondre aux requêtes entrantes.

8.5.1 Connexion à l'IHM

À la suite de la création d'une connexion spécifique pour l'IHM, définie dans la base de données Optiplant, il a été possible via CODESYS d'ouvrir une connexion sur la base de données à partir de l'IHM. Des requêtes d'insertion sur la base de données et des requêtes de sélection d'informations ont été vérifiées.

8.6 Documentation et support

Pour faciliter la maintenance et l'utilisation du système, une documentation détaillée a été rédigée. Cette documentation inclut les étapes d'installation et de configuration, les commandes utiles pour gérer les services, et les informations de connexion pour la base de données et phpMyAdmin. Cette documentation a été partagée avec les utilisateurs pour leur permettre de résoudre les problèmes courants et de configurer leur environnement si nécessaire.

9. Conclusion

Ce projet d'ombrière automatisée a permis d'aboutir à un système complet et fonctionnel, marquant une évolution significative par rapport au travail initié lors du semestre précédent. L'ensemble des fonctionnalités développées - incluant une interface web de supervision, une base de données relationnelle Optiplant performante, une API météo intégrée et un système de Livestream temps réel - forme désormais une solution cohérente et opérationnelle.

La mise en œuvre d'un système de gestion des utilisateurs avec droits différenciés a apporté une dimension essentielle de sécurité et de personnalisation de l'accès aux données. L'adoption rigoureuse d'outils professionnels comme Git/GitHub pour le versionning du code, couplée à une méthodologie SCRUM adaptée, a structuré efficacement le travail d'équipe tout au long du développement. Le déploiement réussi sur plateforme LattePanda, avec la configuration minutieuse des services nécessaires (MySQL, Apache, phpMyAdmin) et l'ouverture des ports appropriés, a concrétisé la viabilité technique du système.

Ce projet a représenté une opportunité unique pour chaque membre de l'équipe de consolider et d'approfondir des compétences clés en programmation web avancée, conception et gestion de bases de données complexes, ainsi qu'en intégration système. Le respect strict des normes d'accessibilité (RGAA) a par ailleurs démontré notre capacité à prendre en compte les exigences réglementaires dans une solution technique.

Au-delà de la réalisation technique, ce travail a prouvé la faisabilité d'un système automatisé intelligent, parfaitement supervisable à distance, répondant précisément aux objectifs initiaux tout en ouvrant des perspectives concrètes pour des applications industrielles dans le domaine de l'agriculture connectée et de la gestion énergétique optimisée. La robustesse de la solution développée et la méthodologie employée témoignent de notre capacité à mener à bien des projets complexes intégrant à la fois des composants matériels et logiciels.